



UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/748,029	12/22/2000	Ariel Cohen	00-177	4217
24319	7590	08/11/2005	EXAMINER	
LSI LOGIC CORPORATION 1621 BARBER LANE MS: D-106 MILPITAS, CA 95035			LI, AIMEE J	
			ART UNIT	PAPER NUMBER
			2183	

DATE MAILED: 08/11/2005

Please find below and/or attached an Office communication concerning this application or proceeding.

Office Action Summary

Application No.

09/748,029

Applicant(s)

COHEN ET AL.

Examiner

Aimee J. Li

Art Unit

2183

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --
Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE 3 MONTH(S) FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

- 1) ☒ Responsive to communication(s) filed on 16 May 2005 and 18 May 2005.
- 2a) ☒ This action is **FINAL**. 2b) ☐ This action is non-final.
- 3) ☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

- 4) ☒ Claim(s) 1-25 is/are pending in the application.
- 4a) Of the above claim(s) _____ is/are withdrawn from consideration.
- 5) ☐ Claim(s) _____ is/are allowed.
- 6) ☒ Claim(s) 1-23 and 25 is/are rejected.
- 7) ☒ Claim(s) 24 is/are objected to.
- 8) ☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

- 9) ☐ The specification is objected to by the Examiner.
- 10) ☐ The drawing(s) filed on _____ is/are: a) ☐ accepted or b) ☐ objected to by the Examiner.
Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).
Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).
- 11) ☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

- 12) ☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).
a) ☐ All b) ☐ Some * c) ☐ None of:
1. ☐ Certified copies of the priority documents have been received.
 2. ☐ Certified copies of the priority documents have been received in Application No. _____.
 3. ☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).
- * See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

- | | |
|--|---|
| 1) <input checked="" type="checkbox"/> Notice of References Cited (PTO-892) | 4) <input type="checkbox"/> Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ |
| 2) <input type="checkbox"/> Notice of Draftsperson's Patent Drawing Review (PTO-948) | 5) <input type="checkbox"/> Notice of Informal Patent Application (PTO-152) |
| 3) <input type="checkbox"/> Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)
Paper No(s)/Mail Date _____ | 6) <input type="checkbox"/> Other: _____ |

DETAILED ACTION

1. Claims 1-23 have been considered. Claims 1, 2, 4-5, 8, 16-18, and 20-23 have been amended as per Applicant's request. New Claims 24-25 have been added as per Applicant's request.

Papers Submitted

2. It is hereby acknowledged that the following papers have been received and placed on record in the file: Amendment as filed 16 May 2005 and Declaration under 37 C.F.R. §1.132 as filed 18 May 2005.

Claim Rejections - 35 USC § 102

3. The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

4. Claims 1-7, 9-10, 16, 18-19, 23 and 25 are rejected under 35 U.S.C. 102(e) as being anticipated by Patel et al., U.S. Patent No. 6,332,215.

5. Regarding claim 1, Patel has taught an apparatus comprising:

a. A processor (20 of Fig.1):

i. Comprising a number of internal general purpose registers (46 and 48 of Fig.3),

- ii. Configured to manipulate contents of said internal general purpose registers in response to instruction codes of a first instruction set (see Col.4 lines 6-22),
- b. A processor interface circuit coupled to said processor (28 and 25 of Fig. 3);
- c. A memory interface circuit coupled to a memory device (24 of Fig. 3);
- d. An extension stack coupled between said processor interface and said memory interface (50 of Fig.3) and configured to (i) receive data from and present data to said memory interface circuit (52 and 42 of Fig. 3) and (ii) receive data from and present data to said processor interface circuit (Fig. 1),
- e. A translator circuit (see Fig.3)
 - i. Coupled between said processor interface and said memory interface (42 of Fig. 3)
 - ii. Configured to implement a stack using one or more of said internal general purpose registers of said processor and said extension stack (see Col.4 lines 9-22). Here, the JVM stack is implemented using both the internal general purpose registers (46 and 48 of Fig.3) and the extension stack (50 of Fig.3), as not only are the java registers in the internal registers stored within the extension stack (see Col.4 lines 50-54), but the internal registers contain the pointers to the extension stack, allowing data to be written to and read from the stack (see Col.4 lines 12-17).

Art Unit: 2183

6. Regarding claim 2, Patel has taught the apparatus according to claim 1, wherein said one or more internal general purpose registers are used to store a top of said stack (see Col.4 lines 8-17).

7. Regarding claim 3, Patel has taught the apparatus according to claim 2, wherein said top of said stack comprises a Java virtual machine (JVM) top of stack (TOS) (see Col.4 lines 8-17).

8. Regarding claim 4, Patel has taught the apparatus according to claim 1, wherein said internal general purpose registers are dynamically allocated in response to stack status (see Col.5 line 51 – Col.6 line 11).

9. Regarding claim 5, Patel has taught the apparatus according to claim 1, wherein said translator circuit is further configured to generate one or more instruction codes of the first instruction set for controlling the internal general purpose registers in response to an instruction code of a second instruction set (see Col.4 lines 6-22 and Col.5 lines 48-60). Here, an instruction of the second instruction set (iadd) is translated into an instruction of the first instruction set (ADD R1, R2), and the ADD instruction controls and manipulates the internal registers by reading them, performing an operation on them, and then writing back a result to one of the registers, wherein all of the registers involved are on the operand stack (see Col.5 lines 57-60), which is implemented using the internal registers (44 of Fig.3) and the extension stack (50 of Fig.3) (see Col.4 lines 17-22).

10. Regarding claim 6, Patel has taught the apparatus according to claim 5, wherein said instruction code of said second instruction set comprises a stack instruction (see Col.2 lines 19-27). It is well known in the art that Java Virtual Machine instructions are stack-based, with their operands being provided on a stack rather than in registers. It is therefore inherent that any non-

Art Unit: 2183

trivial instructions of the stack-based Java instruction set (first instruction set), such as the “iadd” instruction, will be “stack operations” due to their stack-based nature.

11. Regarding claim 7, Patel has taught the apparatus according to claim 1, wherein said translator circuit comprises a stack management unit (64 of Fig.3) coupled to said processor interface, said memory interface, and said extension stack (see Col.5 lines 6-11).

12. Regarding claim 9, Patel has taught the apparatus according to claim 1, wherein said extension stack (50 of Fig.3) is implemented as a last-in-first-out (LIFO) memory. While it is not taught explicitly, it is inherent and is well known in the art that stacks are implemented as last-in first-out data structures.

13. Regarding claim 10, Patel has taught the apparatus according to claim 1, wherein said extension stack comprises both head (see “Optop”, Col.4 lines 14-15) and tail interfaces (see “Frame”, Col.4 lines 15-16).

14. Regarding claim 16, Patel has taught the apparatus according to claim 7, wherein said stack management unit (64 of Fig.3) is configured to control:

- a. Pushes to said one or more internal general purpose registers from said extension stack (see Col.5 lines 6-11), and
- b. Pops from said one or more internal general purpose registers to said extension stack (Col.5 lines 6-11).

15. Regarding claim 18, Patel has taught a method for implementing a Java virtual machine top of stack comprising the steps of:

Art Unit: 2183

- a. Translating one or more instruction codes of a first instruction set into sequences of instruction codes of a second instruction set (see Col.3 lines 50-53 and Col.4 lines 1-4),
 - b. Manipulating contents of one or more internal general purpose registers (46 and 48 of Fig.3) of a processor in response to said sequence of instruction codes of said second instruction set (see Col.4 lines 6-22),
 - c. Implementing a stack comprising said one or more internal general purpose registers and an extension stack coupled between said processor and a memory device (50 of Fig.3), wherein said one or more internal general purpose registers (44 of Fig.3) are configured as a top of stack (see Col.4 lines 9-22) and said extension stack is configured to (i) receive data from and present data to said memory interface circuit (52 and 42 of Fig. 3) and (ii) receive data from and present data to said processor interface circuit (Fig. 1). Here, the JVM stack is implemented using both the internal registers (46 and 48 of Fig.3) and the extension stack (50 of Fig.3), as not only are the java registers in the internal registers stored within the extension stack (see Col.4 lines 19-22), but the internal registers contain the pointers to the extension stack, allowing data to be written to and read from the stack (see Col.4 lines 12-17).
16. Regarding claim 19, Patel has taught the method according to claim 18, wherein said instruction codes of said first instruction set comprise stack operations (see Col.2 lines 19-27). It is well known in the art that Java Virtual Machine instructions are stack-based, with their operands being provided on a stack rather than in registers. It is therefore inherent that any non-

Art Unit: 2183

trivial instructions of the stack-based Java instruction set (first instruction set) will be “stack operations” due to their stack-based nature.

17. Regarding claim 23, Patel has taught the apparatus according to claim 1, further comprising:

- a. A register block (44 of Fig.3) coupled between said processor interface circuit and said extension stack and configured to operate as a bridge between said processor and said extension stack (see Fig.3).

18. Regarding claim 25, Patel has taught wherein said translator circuit is further configured to extend said stack into said memory device (see Col. 4 lines 50-54).

Claim Rejections - 35 USC § 103

19. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

(a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negated by the manner in which the invention was made.

20. Claims 14 and 15 are rejected under 35 U.S.C. 103(a) as being unpatentable over Patel et al, U.S. Patent No. 6,332,215.

21. Regarding claim 14, Patel has taught the apparatus according to claim 7, but has not explicitly taught wherein said stack management unit is further configured to track which internal registers are used for the stack.

22. However, Patel has taught the tracking of another set of registers usage in a different stack (see Col.5 lines 43-47) so that it can be determined which operands are valid operands and thus which can be used in handling overflow/underflow (see Col.5 line 61 – Col.6 line 11).

Art Unit: 2183

Because Patel has not taught explicit reasons why the same tracking of registers can't be used on the internal registers (44 of Fig.3) and the extension stack (50 of Fig.3) that are used to implement the stack, one of ordinary skill in the art would have found it obvious to modify Patel to further track which registers of the internal registers (44 of Fig.3) are used to implement the stack so that valid operands can be identified such that overflow/underflow situations can be handled correctly.

23. Regarding claim 15, Patel has taught the apparatus according to claim 14, wherein said stack management unit is further configured to track how many internal registers are used for the stack (see Patel, Col.4 lines 8-22 and Col.5 lines 34-37). Here, the operand stack is implemented using the Java registers (44 of Fig.3) and the java stack (50 of Fig.3), and counters keep track of how many entries have been placed on the operand stack (see Col.5 lines 34-37).

24. Claims 8, 11-13, 17 and 20-22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Patel et al, U.S. Patent No. 6,332,215 as applied to claim 7 above, and further in view of Tremblay et al, U.S. Patent No. 6,021,469.

25. Regarding claim 8, Patel has taught the apparatus according to claim 7, wherein said stack management unit (64 of Fig.3) is configured to control transfers between:

- a. Said internal general purpose registers and said extension stack (see Patel, Col.5 lines 6-11).

26. Patel has not explicitly taught wherein the stack management unit is configured to control transfers between the extension stack and said memory device.

27. However, Tremblay has taught the transferring of data from a stack to a data cache so that pipeline stalls due to overflow/underflow conditions can be avoided (see Tremblay, Col.18

Art Unit: 2183

lines 46-61). Therefore, one of ordinary skill in the art would have found it obvious to modify Patel such that the stack management unit could control transfers between the extension stack and a data cache so that costly pipeline stalls will be avoided.

28. Regarding claim 11, Patel in view of Tremblay has taught the apparatus according to claim 8, wherein said extension stack:

- a. Is emptied to said memory device to prevent an overflow (see above paragraphs 26-28 and Tremblay, Col.18 lines 46-61), and
- b. Filled from said memory device to prevent an underflow (see above paragraphs 26-28 and Tremblay, Col.18 lines 46-61).

29. Regarding claim 12, Patel in view of Tremblay has taught the apparatus according to claim 11, wherein said memory device comprises a main memory of said processor (see above paragraphs 26-29 and Tremblay, Col.18 lines 34-41).

30. Regarding claim 13, Patel has taught the apparatus according to claim 7, but has not explicitly taught wherein said extension stack is configured to indicate an almost empty or almost full condition.

31. However, Tremblay has taught the use of high and low watermarks to indicate when the stack is near full or empty so to avoid stalling the pipeline due to overflows or overwrites (see Tremblay, Col.18 lines 46-61). Therefore, it would have been obvious to modify Patel to include high and low watermarks as Tremblay does to more clearly indicate when a potential overflow or overwrite condition occurs and avoid costly pipeline stalls.

32. Regarding claim 17, Patel has taught an apparatus comprising:

Art Unit: 2183

- a. Means for manipulating data in response to instruction codes of a first instruction set (see Col.4 lines 6-22), said manipulating means comprising a number of internal general purpose registers (46 and 48 of Fig.3),
- b. Means for translating instruction codes of a second instruction set into sequences of said instruction codes of a said first instruction set, wherein said translating means is configured to
 - i. Implement a stack (see Col.4 lines 9-22) with one or more of said internal general purpose registers (46 and 48 of Fig.3) and an extension stack coupled between said manipulating means and a memory device (50 of Fig.3). Here, the JVM stack is implemented using both the internal registers (46 and 48 of Fig.3) and the extension stack (50 of Fig.3), as not only are the java registers in the internal registers stored within the extension stack (see Col.4 lines 19-22 and 50-54), but the internal registers contain the pointers to the extension stack, allowing data to be written to and read from the stack (see Col.4 lines 12-17).
 - ii. Use said one or more of said internal general purpose registers as a top of stack (see Col.4 lines 8-17),
 - iii. Transfer contents of said one or more internal general purpose registers to said extension stack (see Col.5 lines 6-11),
 - iv. Transfer contents of said extension stack to said one or more internal general purpose registers (see Col.5 lines 6-11).

Art Unit: 2183

33. Patel has not explicitly taught wherein the “stack” can empty to or refill from a memory device.

34. However, Tremblay has taught the emptying and refilling of data between a stack and a memory device so that pipeline stalls due to overflow/underflow conditions can be avoided (see Tremblay, Col.18 lines 46-61). Therefore, one of ordinary skill in the art would have found it obvious to modify Patel such that the implemented stack could empty to and refill from a memory device so that costly pipeline stalls will be avoided.

35. Regarding claim 20, Patel has taught the method according to claim 18, further comprising the step of:

- a. Transferring values between said internal general purpose registers and said extension stack (see Col.5 lines 6-11) in response to a first one or more of said sequences of instruction codes of said second instruction set (see Col.4 lines 6-22),

36. Patel has not explicitly taught the transferring of values between the extension stack and said memory device in response to watermark indications from the extension stack.

37. However, Tremblay has taught the use of high and low watermarks to indicate when a stack is near full or empty so to avoid stalling the pipeline due to overflows or overwrites (see Tremblay, Col.18 lines 46-61). Therefore, it would have been obvious to modify Patel to include high and low watermarks and transfer values between the stack and a memory device as Tremblay does to more clearly indicate when a potential overflow or overwrite condition will occur and prevent it, thus avoiding costly pipeline stalls.

Art Unit: 2183

38. Regarding claim 21, Patel has taught the method according to claim 18, further comprising the step of:

- a. Generating control signals configured to:
 - i. Transfer values from said one or more internal general purpose registers to said extension stack (see Col.5 lines 6-11),
 - ii. Restore values from said extension stack to said one or more internal general purpose registers (see Col.5 lines 6-11).

39. Patel has not explicitly taught the extension stack emptying to and refilling from said memory device in response to watermark levels.

40. However, Tremblay has taught the use of high and low watermarks to indicate when a stack is near full or empty so to avoid stalling the pipeline due to overflows or overwrites (see Tremblay, Col.18 lines 46-61). Therefore, it would have been obvious to modify Patel to include high and low watermarks and transfer values between the stack and a memory device as Tremblay does to more clearly indicate when a potential overflow or overwrite condition will occur and prevent it, thus avoiding costly pipeline stalls.

41. Regarding claim 22, Patel has taught the apparatus according to claim 1, wherein said translator circuit is configured to:

- a. Transfer contents of said one or more internal general purpose registers to said extension stack (see Col.5 lines 6-11),
- b. Transfer contents of said extension stack to said one or more internal general purpose registers (see Col.5 lines 6-11).

Art Unit: 2183

42. Patel has not explicitly taught the extension stack emptying to and refilling from said memory device in response to overflow or underflow conditions, nor has Patel taught the transferring of contents from one or more internal general purpose registers to or from the extension stack in response to overflow or underflow conditions.

43. However, Tremblay has taught the emptying and refilling of data between a stack and a memory device so that pipeline stalls due to overflow/underflow conditions can be avoided (see Tremblay, Col.18 lines 46-61). Therefore, one of ordinary skill in the art would have found it obvious to modify Patel such that the implemented stack could empty to and refill from a memory device so that costly pipeline stalls will be avoided.

44. Furthermore, Tremblay has taught the transferring of contents from one level of a memory hierarchy to another when there is an overflow/underflow condition (see Tremblay, Col.18 lines 27-45) in order to prevent pipeline stalls (see Tremblay, Col.18 lines 46-61). Because the internal registers and extension stack of Patel are a memory hierarchy (see Patel, Fig.3), one of ordinary skill in the art would have found it obvious to transfer contents of the internal registers to and from the external stack upon an overflow/underflow condition so that costly pipeline stalls are prevented.

Allowable Subject Matter

45. Claim 24 is objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims.

Response to Arguments

Art Unit: 2183

46. Applicant's arguments filed 16 May 2005 have been fully considered but they are not persuasive. Since the arguments presented in the Amendment filed 16 May 2005 basically summarize the points made in the Declaration of Steven M. Emerson Pursuant to 37 C.F.R. §1.132, the Examiner will respond to the specific points made by Mr. Emerson and these responses will also be applicable to the arguments in the Amendment utilizing Mr. Emerson's statements as support.

47. Before responding to the arguments presented in the Declaration, the Examiner would like to note that Mr. Emerson is an employee of LSI logic Corporation, the current assignee of the instant application. The MPEP states in 716.01(c) under III. Opinion Evidence

In assessing the probative value of an expert opinion, the examiner must consider the nature of the matter sought to be established, the strength of any opposing evidence, the interest of the expert in the outcome of the case, and the presence or absence of factual support for the expert's opinion.

48. Mr. Emerson has a higher vested interest in the outcome of the case, since, while not a direct inventor, he has been an employee of the assignee for 10 years and he is an Engineering Manager in the field of art the application falls in, i.e. Processor Cores Technology. There is also no external evidence to support Mr. Emerson's statements in the Declaration. The Declaration relies upon Mr. Emerson's own interpretation and opinion of the reference(s) in question without any outside evidence to support his interpretation and opinion. Also, the Declaration is dependent upon the previous interpretation of the claim language, which was prior to the current amendments made. While the same art is still applied to the current claims, the interpretation of the current prior art had to be changed to match the new claim language.

49. The first seven bullets (1-7) and the last bullet (22) are merely summarizing the claims and stating Mr. Emerson's relation to the case, so the Examiner will start responding from bullet

Art Unit: 2183

8, which summarizes Mr. Emerson's stance with "...Patel does not disclose and would not suggest and extension stack as presently claimed" and its supporting bullets.

50. Bullet 9 states

Since Patel shows the JAVA stack 50 only connected to the hardware registers 44, it follows that Patel does not disclose or suggest an extension stack coupled between the processor interface and the memory interface and configured to (i) receive data from and present data to said memory interface circuit and (ii) receive data from and present data to said processor interface circuit...

51. This has not been found persuasive. The claim language states "an extension stack coupled between said processor interface and said memory interface". This argument seems to suggest that the term "coupled" means direct connections between the stack and the memory and processor interface circuits. The term "coupled" just means that there is a connection and that the two elements somehow influence each other (Merriam-Webster Online Dictionary). In Patel, Figure 3 shows the JAVA Accelerator of Figure 1 in more detail (Patel column 3, line 66 to column 4, line 22). In Figure 1, it can be seen that the JAVA Accelerator is between the memory (element 24) and the CPU (element 26). The JAVA stack in Figure 3 (element 50) is part of the JAVA Accelerator. It is connected to the Instruction Translation (element 42) and JAVA Registers (element 44), which are the other components of the JAVA Accelerator. Since all three of these elements (42, 44, and 50) are part of the JAVA Accelerator, then they are all between the memory and the CPU. Patel's Figure 1 shows that the Hardware JAVA Accelerator (element 22) sends data to the CPU via a selector (element 28) and receives data from the CPU via the data bus 30. Figure 1 also shows that data is received from the Instruction Cache 24.

Art Unit: 2183

However, Figure 1 does not show the Instruction Cache received data. Patel's Figure 3 shows this with a connection back from the JAVA registers 44 via a selector (element 52).

52. Bullet 10 states "Patel states that the Java registers are part of the Java Virtual Machine and should not be confused with general registers 46 or 48 which are operated upon by the central processing unit 26..." This has not been found persuasive. This is based upon the Examiner's previous interpretation of the reference when the claim language just stated "internal registers". Now that the language has been clarified to state "internal general purpose registers" the Examiner's stance on the reference has changed. Now Patel's Figure 3, elements 46 and 48 read upon the claim. Patel states in column 4, lines 46-54 that "the Java CPU register file 48, or in an alternate embodiment the *conventional CPU register file 46, can be used to store portions of the operand stack...* (emphasis added)" and the Instruction Translator (element 42) "can operate upon the operand stack and variable values stored in the Java CPU register file 48..." This means that the Java register file 48 stores portions of a stack and these portions can be operated on. Patel also states in column 4, lines 42-49 that the Java CPU register file 48 can be separate or part of the CPU register file 46, i.e. part of the CPU's internal general purpose register file. Also, "general purpose register" in the broadest sense of the term is a register that can be used for different purposes, such as a special handler of data (Rosenberg's Dictionary of Computers, Information Processing & Telecommunications). The Hardware Java Registers (element 44) are then a type of "internal general purpose registers" since they are an internal special handler of the Java data in the Java Accelerator (22 of Fig. 1).

53. Bullet 11 states "Element 44 in FIG. 3 of Patel is described as hardware Java registers that store the Java registers defined by the Java Virtual Machine..." This has not been found

Art Unit: 2183

persuasive. As state above, this is based upon claim and prior art interpretations made prior to the new claim language that has been added. The new language clarifies the registers being referred to and the relationship each element has with the other elements.

54. Bullet 12 states

...a person skilled in the field of the present invention would (i) not consider Element 44 in FIG. 3 of Patel to be internal general purpose registers of the processor 25 of Patel and (ii) not consider the processor 25 of Patel as being configured to manipulate contents of the hardware Java registers 44 of Patel.

55. This has not been found persuasive. As stated above, Mr. Emerson's point was based upon the Examiner's interpretation of the claims and prior art before the new claim language was added to the claims. Also, as stated above, with the newly amended claims, elements 46 and 48 are now the general purpose registers, and as taught by Patel in lines cited above, the registers 46 and 48 are manipulated by the processor.

56. Bullets 13-17 state

13. FIG. 3 of Patel shows data (i.e., output from the instruction cache 24) moving only from the instruction cache 24 to (i) the Java accelerator 42 or (ii) the processor 25.
14. FIG. 3 of Patel shows data (i.e., output from the Java accelerator 42 moving only from the Java accelerator 42 to the processor 24 (i.e., via multiplexer 28).

15. Patel teaches that data (i.e., byte code instructions) is received from the instruction cache 24 in response to the Java PC or the normal program counter 54.
16. A person of ordinary skill in the field of the present invention would consider the Java PC and normal program counter to be addresses used to retrieve data from the instruction cache 24.
17. A person of ordinary skill in the field of the invention would consider using addresses to retrieve data from a memory to be different from presenting data to the memory as presently claimed.

57. This has not been found persuasive. First, to clarify, Patel's Figure 3, element 42 is the Java Accelerator *Instruction Translation unit*, not the entire Java Accelerator, as shown in 22 of Figure 1. The entire Java Accelerator as taught by Patel in column 3, line 66 to column 4, lines 22, as shown in 22 of Figure 1, is the Java Accelerator (42), Hardware Java Registers (44), Java Stack (50), selector (52), and Java Accelerator Controller (64). To equate the Java Accelerator 22 of Figure 1 to only the Java Accelerator Instruction Translation unit 42 of Figure 3 would be incorrect and inaccurate, since there are other pieces of the Java Accelerator 22 of Figure 1 present in Figure 3. The claim language states "receive data from and present data to said memory interface circuit and (ii) receive data from and present data to said processor interface circuit". The claim language does not state what type of data nor the purpose of the data being received from and presented to memory and the CPU. Data, in the broadest meaning of the term, is "a representation of facts, concepts, or instructions in a formalized manner suitable for communication, interpretation, or processing by...automatic means" or "any representations

Art Unit: 2183

such as characters or analog quantities to which meaning is, or might be, assigned (Rosenberg's Dictionary of Computers, Information Processing & Telecommunications).” This means the data presented to and received from the memory and CPU can be numbers, characters, addresses, etc., since they are all data. Addresses, for example, represent the location of an instruction in a manner easy for communication between units. Therefore, the fact that the Java PC and normal program counter are presented from the Java Accelerator to the Instruction cache means that data is being presented to the memory.

58. Bullet 18 states

Since Patel only shows information moving (i) from the instruction cache 24 and (ii) to the processor 25, a person skilled in the field of the presently claimed invention would not consider either Element 44 or Element 50 to be configured to (i) receive data from a processor and (ii) present data to a memory...

59. This has not been found persuasive. As stated above, these elements 44 and 50 are all part of the Java Accelerator 22 of Figure 1, which receives data from the processor, as shown in Figure 1, and presents data to memory, as shown in Figure 3. The entire Java Accelerator as taught by Patel in column 3, line 66 to column 4, lines 22, as shown in 22 of Figure 1, is the Java Accelerator (42), Hardware Java Registers (44), Java Stack (50), selector (52), and Java Accelerator Controller (64). To equate the Java Accelerator 22 of Figure 1 to only the Java Accelerator Instruction Translation unit 42 of Figure 3 would be incorrect and inaccurate, since there are other pieces of the Java Accelerator 22 of Figure 1 present in Figure 3.

60. Bullet 19 states “...does not disclose and would not suggest to one of ordinary skill in the field of the present invention an extension stack comprises both head and tail interfaces...” This

Art Unit: 2183

has not been found persuasive. Patel in column 4, lines 14-15 describe the "Optop" pointer, which is to the top of the stack, i.e. the head of the stack. Patel describes in column 4, lines 15-16 the "Frame" pointer, which is "to the execution environment of the current method". He execution environment means where the sequence of bits are held, i.e. where the beginning (tail) and the end (head) of the sequence is. The definition of "Frame", when applied to data transmission, is "the sequence of contiguous bits bracketed by and including beginning and ending flag sequences". Patel's Java Accelerator transmits the data from the stack to help with processing of the Java byte codes.

61. Bullet 20 states "...a person of ordinary skill in the field of the invention would understand the hardware Java stack 50 as being an alternative to storing the stack in the CPU associated register files." This has not been found persuasive. Patel states later in the description in column 4, lines 46-49 "As described below with respect to FIGS. 3 and 4, the Java CPU register file 48, or in an alternate embodiment the conventional CPU register file 46, can be used to store *portions of the operand stack* and some of the variables (emphasis added)." This means that the Java stack 50 is still needed to store the entire stack, since the register file only contains portions of the stack, not the full stack.

62. Bullet 21 states

The position taken in the Office Action...that the element 44 in FIG. 3 of Patel correspond to both (i) the presently claimed internal general purpose registers and (ii) the presently claimed register block coupled between the processor interface circuit and the extension stack are not supported by the disclosure of Patel.

Art Unit: 2183

63. This has not been found persuasive. Again, this is based upon the old rejection, which was before the claim language had been amended to include the “internal general purpose registers” and the receiving from and presenting to memory and the CPU. As is evidenced by the responses to the previous arguments above, the interpretation of the claims and the prior art in relation to the claims had to be adjusted to incorporate the amendments made to the claim. For purposes of the arguments, the “internal general purpose registers” have been equated to register files 46 and 48 of Figure 3. The Hardware Java Registers 44 of Figure 3 are still equated to the register block coupled between the processor interface and the extension stack. As stated above, element 44 is part of the Java Accelerator, which is connected to the CPU via element 28. Element 44 is also connected between Java Stack 50 and the other elements of the Java Accelerator 22. So, element 44 must be connected between the CPU 25 and the Java Stack 50.

Conclusion

64. **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth in 37 CFR 1.136(a).

65. A shortened statutory period for reply to this final action is set to expire **THREE MONTHS** from the mailing date of this action. In the event a first reply is filed within **TWO MONTHS** of the mailing date of this final action and the advisory action is not mailed until after the end of the **THREE-MONTH** shortened statutory period, then the shortened statutory period will expire on the date the advisory action is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of the advisory action. In no event, however, will the statutory period for reply expire later than **SIX MONTHS** from the mailing date of this final action.

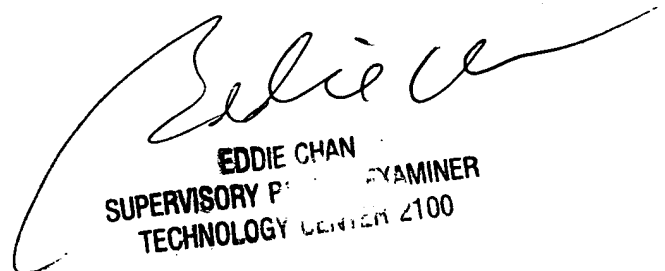
Art Unit: 2183

66. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Aimee J Li whose telephone number is (571) 272-4169. The examiner can normally be reached on M-T 7:30am-5:00pm.

67. If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Eddie Chan can be reached on (571) 272-4162. The fax phone number for the organization where this application or proceeding is assigned is 703-872-9306.

68. Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see <http://pair-direct.uspto.gov>. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

AJL
Aimee J. Li
7 January 2005



EDDIE CHAN
SUPERVISORY P... EXAMINER
TECHNOLOGY CENTER 2100